Assignment 1 - written parts

Tuesday, September 15, 2020 10:01 PM

1. derivative @ $(x \pm dx)$

$$f_1'(x) = \frac{f(x+dx) - f(x-dx)}{2dx}$$

g_i: g_0: order unity random #'s; ε: machine precision

$$f(x+dx) = f(x) + dx f'(x) + \frac{dx^2 f''(x)}{2} + \frac{dx^3 f'''(x)}{6} + \frac{dx^4 f^4(x)}{24} + \frac{dx^5 f^5(x)}{120} + \dots + g_1 \varepsilon f(x)$$

$$f(x-dx) = f(x) - dx f'(x) + \frac{dx^2 f''(x)}{2} - \frac{dx^3 f'''(x)}{6} + \frac{dx^4 f^4(x)}{24} - \frac{dx^5 f^5(x)}{120} + \dots + g_0 \varepsilon f(x)$$

$$f_1'(x) = \frac{2dx f'(x) + (1/3) dx^3 f'''(x) + (1/60) dx^5 f^5(x) + \dots + (g_1 - g_0)\varepsilon f(x)}{2dx}$$

$$f'(x) = f'(x) + \frac{1}{6} dx^2 f'''(x) + \frac{1}{120} dx^4 f^5(x) + \dots + \frac{g \varepsilon f(x)}{2dx} \qquad g = g_1 - g_0$$

derivative at $x \pm 2dx$:

$$f_2' = \frac{f(x+2dx) - f(x-2dx)}{4dx}$$

$$f(x+2dx) = f(x) + 2dx f'(x) + \frac{4dx^2 f''(x)}{2} + \frac{8dx^3 f'''(x)}{6} + \frac{16 dx^4 f^4(x)}{24} + \frac{32 dx^5 f^5(x)}{120} + \dots + g_2 \varepsilon f(x)$$

$$f(x-2dx) = f(x) - 2dx f'(x) + \frac{4dx^2 f''(x)}{2} - \frac{8 dx^3 f'''(x)}{6} + \frac{16 dx^4 f^4(x)}{24} - \frac{32 dx^5 f^5(x)}{120} + \dots + g_3 \varepsilon f(x)$$

$$f_2'(x) = \frac{4dx f'(x) + (8/3) dx^3 f'''(x) + (8/15) dx^5 f^5(x) + \dots + (g_2 - g_3)\varepsilon f(x)}{4dx}$$

$$f_2'(x) = f'(x) + \frac{2}{3} dx^2 f'''(x) + \frac{2}{15} dx^4 f^5(x) + \dots + \frac{g' \varepsilon f(x)}{4dx} \qquad g' = (g_2 - g_3)$$

To eliminate third-order term:

$$f'(x) = \frac{4 f_1'(x) - f_2'(x)}{3} = \frac{4f'(x) - f'(x) + (2/3) dx^2 f'''(x) - (2/3) dx^2 f'''(x) + (1/30) dx^4 f^5(x) - (2/15) dx^4 f^5(x) + \dots + (1/dx)(4g - 1/4\, g')\varepsilon f(x)}{3}$$

$$\boxed{f'(x) = f'(x) - \frac{1}{10} dx^4 f^5(x) + \dots + \frac{1}{dx} g_f \varepsilon f(x)} \qquad g_f = (4g - 1/4\, g')$$

minimize error to find best dx

$$\frac{\partial \text{error}}{\partial dx} = -\frac{4}{10} dx^3 f^5(x) - \frac{g_f \varepsilon f(x)}{dx^2} = 0$$

$$-\frac{2}{5} dx^3 f^5(x) = \frac{g_f \varepsilon f(x)}{dx^2}$$

$$dx^5 = -\frac{5 g_f \varepsilon f(x)}{2 f^5(x)}$$

$$\boxed{dx = \sqrt[5]{\frac{5 g_f \varepsilon f(x)}{2 f^5(x)}}} \qquad f^5(x) \text{ is the 5th derivative of } f(x)$$

$$dx^5 = -\frac{5g\epsilon f(x)}{2f^5(x)}$$

$$dx = -\sqrt[5]{\frac{5g\epsilon f(x)}{2f^5(x)}}$$, where $f^5(x)$ is the 5^{th} derivative of $f(x)$, ϵ is the machine precision, g_ϵ is some random number of order unity

for double precision: $\epsilon \sim 10^{-16}$

then, assuming $f(x)$ and $f^5(x)$ have similar orders of magnitude,

$$|dx| \sim (10^{-16})^{1/5} \sim 10^{-3}$$

as dx gets smaller than that, error should not get better.

Ex:

$$f(x) = e^x \rightarrow f^5(x) = e^x \rightarrow dx = -\sqrt[5]{\frac{5g\epsilon}{2}} \approx 10^{-3.2}$$

$$f(0.01x) = e^{0.01x} \rightarrow f^5(x) = (0.01)^5 e^{0.01x} \rightarrow dx = -\sqrt[5]{\frac{5g\epsilon}{2}\left(\frac{1}{0.01}\right)^5} \approx \left(\frac{10^{-16}}{10^{-10}}\right)^{1/5} = 10^{-2}$$

Problem 1 Code Output:

dx, approximate derivative, and error in exp(x)
-5.0 2.7182818284769237 1.787858749935367e-11
-4.555555555555555 2.7182818284475636 1.1481482431463519e-11
-4.111111111111111 2.7182818284582324 8.126832540256146e-13
-3.666666666666667 2.718281828459375 3.299582829185965e-13
-3.2222222222222223 2.718281828459258 2.1271873151818e-13
-2.7777777777777777 2.7182818284582937 7.513989430663059e-13
-2.3333333333333335 2.7182818284170067 4.203837278282663e-11
-1.8888888888888893 2.718281825937717 2.5213280352431866e-09
-1.4444444444444446 2.7182816772902227 1.5116882234877949e-07
-1.0 2.7182727567264906 9.071732554488676e-06

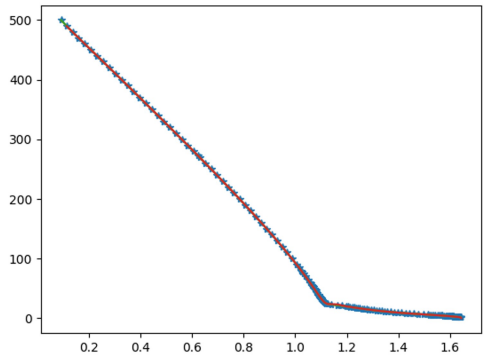
dx, approximate derivative, and error in exp(0.01x)
-5.0 1.0100501670754862 0.9999496654046446
-4.555555555555555 1.010050167088373 0.9999496654175314
-4.111111111111111 1.0100501670838173 0.9999496654129757
-3.666666666666667 1.0100501670839959 0.9999496654131542
-3.2222222222222223 1.010050167084273 0.9999496654134313
-2.7777777777777777 1.010050167083894 0.9999496654130523
-2.3333333333333335 1.0100501670685145 0.9999496653976728
-1.8888888888888893 1.0100501661473054 0.9999496644764637
-1.4444444444444446 1.0100501109133586 0.999949609242517
-1.0 1.0100467962398032 0.9999462945689616

Comments:
For exp(x), the minimum error occurs when the exponent of dx is approximately -3.2, as predicted. Similarly, for exp(0.01), the error is approximately constant as dx shrinks, which makes sense given our calculation of a minimum error when dx is approximately 0.1

Problem 2 Code Output:

Please input the desired voltage:
1.0 (entered by user)

The temperature calculated using the polynomial fit is 92.89954807119172 Kelvin.
And the temperature calculated using the cubic spline fit is 92.8996409775585 Kelvin.
The approximate error in the polynomial fit is 0.05008394883489402
The approximate error in the cubic spline fit is 0.005869676742489251



3. $f(x) = \frac{1}{1+x^2}$

rational function: $y(x) = \frac{p(x)}{q(x)} = \frac{1 + p_1 x + p_2 x^2 + \dots}{1 + q_1 x + q_2 x^2 + \dots}$ + ge y(x)

for fit: $p_1 = p_2 = p_3 = \dots = p_n = 0$

$q_1 = 0,\ q_3 = q_4 = \dots = q_n = 0$

The error should be very small as long as $m \geq 2$.
It should de end almost entirely on the machine precision, which for double precision is $\boxed{10^{-16}}$.

At higher orders, my code agrees with this.

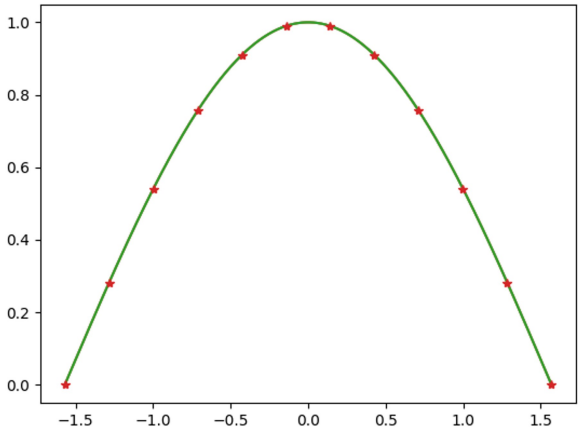
Problem 3 Output:
The error in the polynomial interpolation of cosine is 3.191294279758451e-10
The error in the cubic spline interpolation of cosine is 1.339538690066989e-05
The error in the rational function interpolation of cosine is 9.317498652501655e-09

The error in the polynomial interpolation of the lorentzian is 6.497210473164695e-05
The error in the cubic spline interpolation of the lorentzian is 4.5256398513655765e-05
The error in the rational function interpolation of the lorentzian is 34.32638275518651
The error in the rational function interpolation with np.pinv() is 8.637779136644075e-16

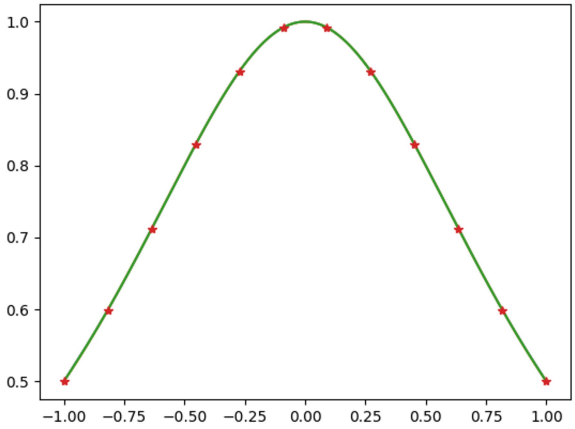
The error in the four-point rational function interpolation of the lorentzian is 9.78238469250138e-17
The error in the four-point rational function interpolation with np.pinv() is 9.78238469250138e-17

Comments:
When you switch from np.linalg.inv() to np.linalg.pinv, it makes no difference for low-order fits. However, as the order increases the determinant of the matrix approaches 0 , so the np.linalg.inv() creates a spike that greatly increases the error while the fit and error using np.linalg.pinv() stays very small. At these higher orders, the p and q calculated using

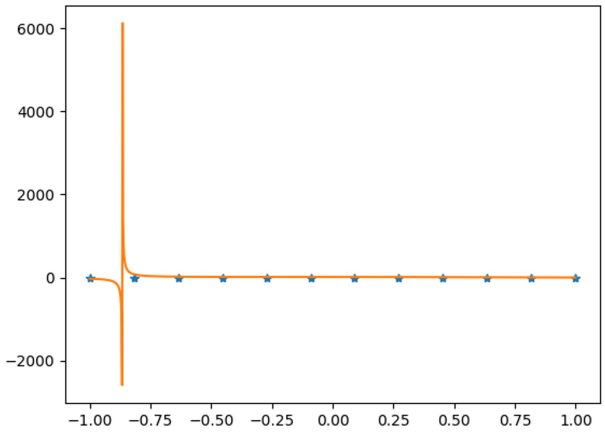
np.linalg.inv() are large because the multiplication factor used to find the inverse (1/determinant) becomes large.



All fits for cosine function (12 points each)

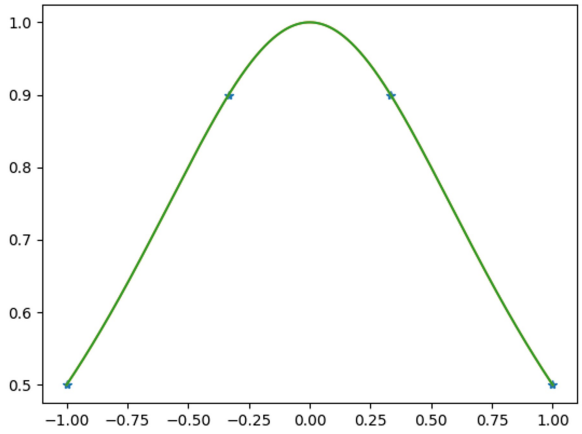


All fits for Lorentzian with 12 points. For the rational function fit, np.linalg.pinv() is used for this higher order fit.



Lorentzian rational function fit using np.inv() at high order (n=6,m=7) Using this command gives a poor fit because the matrix mat has a

determinant very close to zero at high orders, meaning that its inverse values, which are being multiplied by 1/determinant, are very large.



Rational function fits for the Lorentzian with four points each. Both the np.linalg.inv() and the np.linalg.pinv() calculations provide a good fit here, because the higher order terms in the matrix that cause the determinant to approach zero are not included.

Problem 4 Output:

The error in my integrator was approximately 42626.50338994352
And the error in the scipy integrator was approximately 3.1063102137504287e-06

Comments

There is a singularity in the integral at z=R.
My integrator cannot handle this singularity, as it leads to an infinite loop when trying to calculate the integral at z=R. Therefore, the point z=R was excluded from calculating the integral using my integrator. However, the scipy.integrate.quad had no problems.